Variational Inference and Learning II

Modern VI and Variational Autoencoders

Michael U. Gutmann

Probabilistic Machine Learning (INFR11298)
School of Informatics, The University of Edinburgh

Autumn Semester 2026

Assumptions

- Model: $p(\mathbf{v}, \mathbf{h}; \theta)$
- Data: $\mathcal{D} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$, $\mathbf{v}_i \overset{\text{iid}}{\sim} p_*$
- The model is a latent variable model: we have observations for all dimensions of $\mathbf{v}$ but no observations of the latents $\mathbf{h}$.
- For each observation $\mathbf{v}_i$, there is a latent $\mathbf{h}_i$.
- We assume that the pairs $(\mathbf{v}_i, \mathbf{h}_i)$ are iid, so that

$$p(\mathbf{v}_1, \ldots, \mathbf{v}_n, \mathbf{h}_1, \ldots, \mathbf{h}_n; \theta) = \prod_{i=1}^{n} p(\mathbf{v}_i, \mathbf{h}_i; \theta) \tag{1}$$

- We do not deal with the case of unobserved variables due to missing data, i.e. incomplete observations of $\mathbf{v}$. (For VI work on this topic, see e.g. Simkus et al, *Variational Gibbs Inference for Statistical Model Estimation from Incomplete Data*, Journal of Machine Learning Research, 2023)

Program

1. Scalable generic variational learning of latent variable models
2. Deep latent variable models and variational autoencoders

Program

1. Scalable generic variational learning of latent variable models
   - ELBO for iid data
   - Amortised variational inference
   - Reparameterisation and stochastic optimisation

2. Deep latent variable models and variational autoencoders

Lower bound on the likelihood for iid data

- In the ELBO $\mathcal{L}_{\mathbf{x}}(q) = \mathbb{E}_{q(\mathbf{y}|\mathbf{x})}\left[\log p(\mathbf{x}, \mathbf{y}) - \log q(\mathbf{y}|\mathbf{x})\right]$, substitute

$$\mathbf{x} \to (\mathbf{v}_1, \ldots, \mathbf{v}_n), \quad \mathbf{y} \to (\mathbf{h}_1, \ldots, \mathbf{h}_n), \quad p(\mathbf{x}, \mathbf{y}) \to \prod_{i=1}^{n} p(\mathbf{v}_i, \mathbf{h}_i; \boldsymbol{\theta})$$

- Since the true conditional factorises, we use one conditional variational distribution $q(\mathbf{h}|\mathbf{v})$:

$$q(\mathbf{h}_1, \ldots, \mathbf{h}_n|\mathbf{v}_1, \ldots, \mathbf{v}_n) = \prod_{i=1}^{n} q(\mathbf{h}_i|\mathbf{v}_i) \tag{2}$$

- Hence $\mathcal{L}_{\mathcal{D}}$ for iid data $\mathcal{D} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ is a sum of per data-point ELBOs $\mathcal{L}_i$:

$$\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, q) = \sum_{i=1}^{n} \mathcal{L}_i(\boldsymbol{\theta}, q), \quad \mathcal{L}_i(\boldsymbol{\theta}, q) = \mathbb{E}_{q(\mathbf{h}_i|\mathbf{v}_i)}\left[\log \frac{p(\mathbf{v}_i, \mathbf{h}_i; \boldsymbol{\theta})}{q(\mathbf{h}_i|\mathbf{v}_i)}\right]$$

Lower bound on the likelihood for iid data

- From the basic properties of the ELBO, we have

$$\mathcal{L}_i(\boldsymbol{\theta}, q) = \log p(\mathbf{v}_i; \boldsymbol{\theta}) - \mathrm{KL}(q(\mathbf{h}_i|\mathbf{v}_i)||p(\mathbf{h}_i|\mathbf{v}_i; \boldsymbol{\theta})) \quad (3)$$

- Hence, with $\ell(\boldsymbol{\theta}) = \sum_i \log p(\mathbf{v}_i; \boldsymbol{\theta})$,

$$\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, q) = \ell(\boldsymbol{\theta}) - \sum_{i=1}^{n} \mathrm{KL}(q(\mathbf{h}_i|\mathbf{v}_i)||p(\mathbf{h}_i|\mathbf{v}_i; \boldsymbol{\theta})) \quad (4)$$

- If q is unrestricted, the KL terms can be made zero, and we have

$$\max_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) = \max_{\boldsymbol{\theta}, q} \mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, q) \quad (5)$$

- If q is restricted to a family $\mathcal{Q}$ that does not include the true conditional, the KL terms stay positive, and we maximise a lower bound instead:

$$\max_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) > \max_{\boldsymbol{\theta}, q \in \mathcal{Q}} \mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, q) \quad (6)$$

Key technical issues

- We have to maximise $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, q) = \sum_{i=1}^{n} \mathcal{L}_i(\boldsymbol{\theta}, q)$ with respect to $\boldsymbol{\theta}$ and the conditional $q(\mathbf{h}|\mathbf{v})$ where

$$\mathcal{L}_i(\boldsymbol{\theta}, q) = \mathbb{E}_{q(\mathbf{h}_i|\mathbf{v}_i)}\left[\log \frac{p(\mathbf{v}_i, \mathbf{h}_i; \boldsymbol{\theta})}{q(\mathbf{h}_i|\mathbf{v}_i)}\right] \tag{7}$$

  Analytical closed form expressions rarely available.

- We do not want to restrict the model class but solve the optimisation problem for large n and generic $p(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta})$.
- Key technical issues are:

  Issue 1: Learning of conditional variational distribution $q(\mathbf{h}|\mathbf{v})$
  Issue 2: Maximisation when the objective involves the $\mathbb{E}_{q(\mathbf{h}_i|\mathbf{v}_i)}$

Issue 1: Learning the conditional variational distribution

- Learning the conditional $q(\mathbf{h}|\mathbf{v})$ is hard since we have to effectively learn infinitely many pdfs/pmfs (one for each $\mathbf{v}$!).
- But $\mathcal{L}_i$ only involves $q(\mathbf{h}_i|\mathbf{v}_i)$. Hence we can maximise $\mathcal{L}_\mathcal{D}$ by maximising each $\mathcal{L}_i$ separately with respect to $q(\mathbf{h}_i|\mathbf{v}_i)$.
- For fixed $\mathbf{v}_i$, $q(\mathbf{h}_i|\mathbf{v}_i)$ is just a distribution over $\mathbf{h}_i$. We choose it from a parametric family $\mathcal{Q} = \{q(\mathbf{h}; \boldsymbol{\lambda})\}_{\boldsymbol{\lambda}}$, with its own parameter value $\boldsymbol{\lambda}_i$ for each data point:

$$q(\mathbf{h}_i|\mathbf{v}_i) = q(\mathbf{h}_i; \boldsymbol{\lambda}_i)$$

- Writing $q_{\boldsymbol{\lambda}}(\mathbf{h})$ for $q(\mathbf{h}; \boldsymbol{\lambda})$, we need

$$\boldsymbol{\lambda}_i^* = \operatorname*{argmax}_{\boldsymbol{\lambda}} \mathcal{L}_i(\boldsymbol{\theta}, q_{\boldsymbol{\lambda}}) \quad \text{for } i = 1, \ldots, n \tag{8}$$

  The optimal $\boldsymbol{\lambda}_i^*$ depends on $\mathbf{v}_i$ through $\mathcal{L}_i$.
- No closed form solution in general: iterative optimisation for every data point, which is too costly unless n is very small.

Amortisation

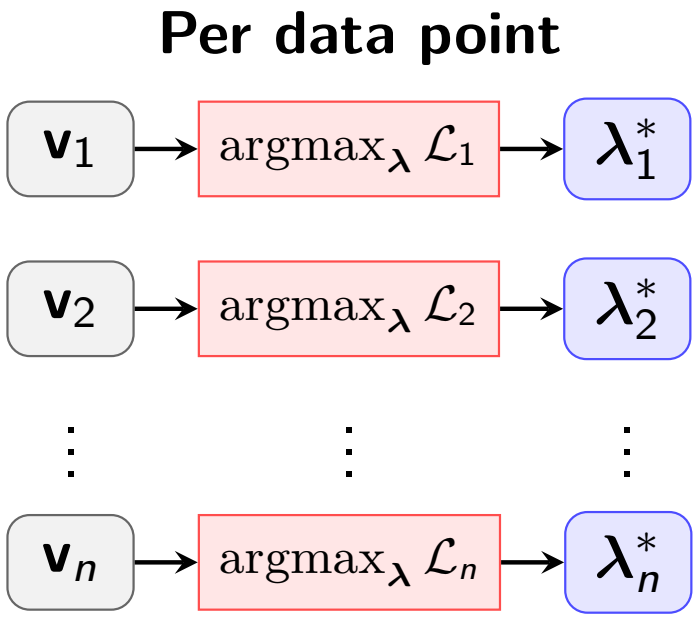
- Instead of optimising each $\boldsymbol{\lambda}_i$ separately, we can learn a function that maps $\mathbf{v}$ to its parameter value $\boldsymbol{\lambda}$:

$$q(\mathbf{h}|\mathbf{v}) = q_{\boldsymbol{\phi}}(\mathbf{h}|\mathbf{v}) = q(\mathbf{h}; \boldsymbol{\lambda}_{\boldsymbol{\phi}}(\mathbf{v})), \qquad \boldsymbol{\lambda}_i = \boldsymbol{\lambda}_{\boldsymbol{\phi}}(\mathbf{v}_i) \tag{9}$$

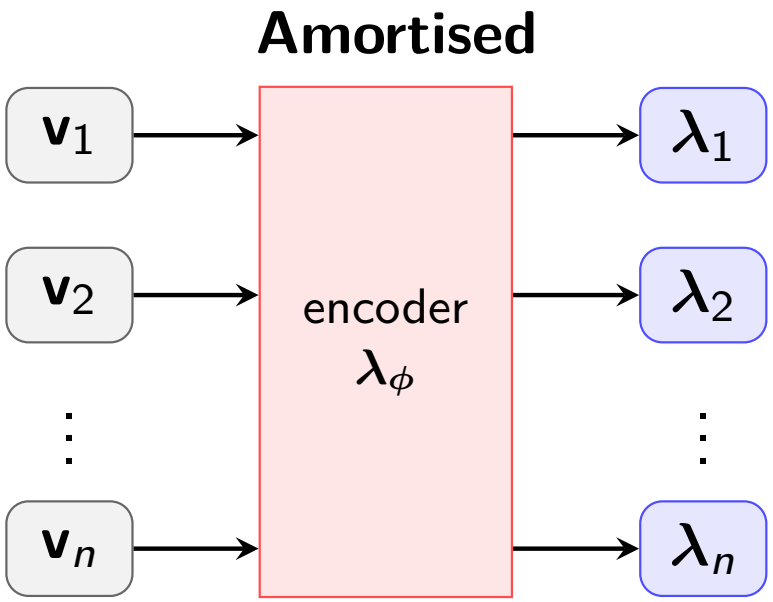
- $\boldsymbol{\lambda}_{\boldsymbol{\phi}}(\mathbf{v})$ is a nonlinear function parameterised by $\boldsymbol{\phi}$. It is called inference or encoder network, or simply encoder.
- Note: $\boldsymbol{\lambda}$ are the parameters of the variational distribution while $\boldsymbol{\phi}$ are the parameters of the encoder network.
- Denote $\mathcal{L}_i(\boldsymbol{\theta}, q_{\boldsymbol{\phi}})$ by $\mathcal{L}_i(\boldsymbol{\theta}, \boldsymbol{\phi})$ and $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, q_{\boldsymbol{\phi}})$ by $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, \boldsymbol{\phi})$.
- We learn $\boldsymbol{\phi}$ (and $\boldsymbol{\theta}$) by maximising

$$\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{i=1}^{n} \mathcal{L}_i(\boldsymbol{\theta}, \boldsymbol{\phi}) \tag{10}$$

Per data-point optimisation vs amortisation



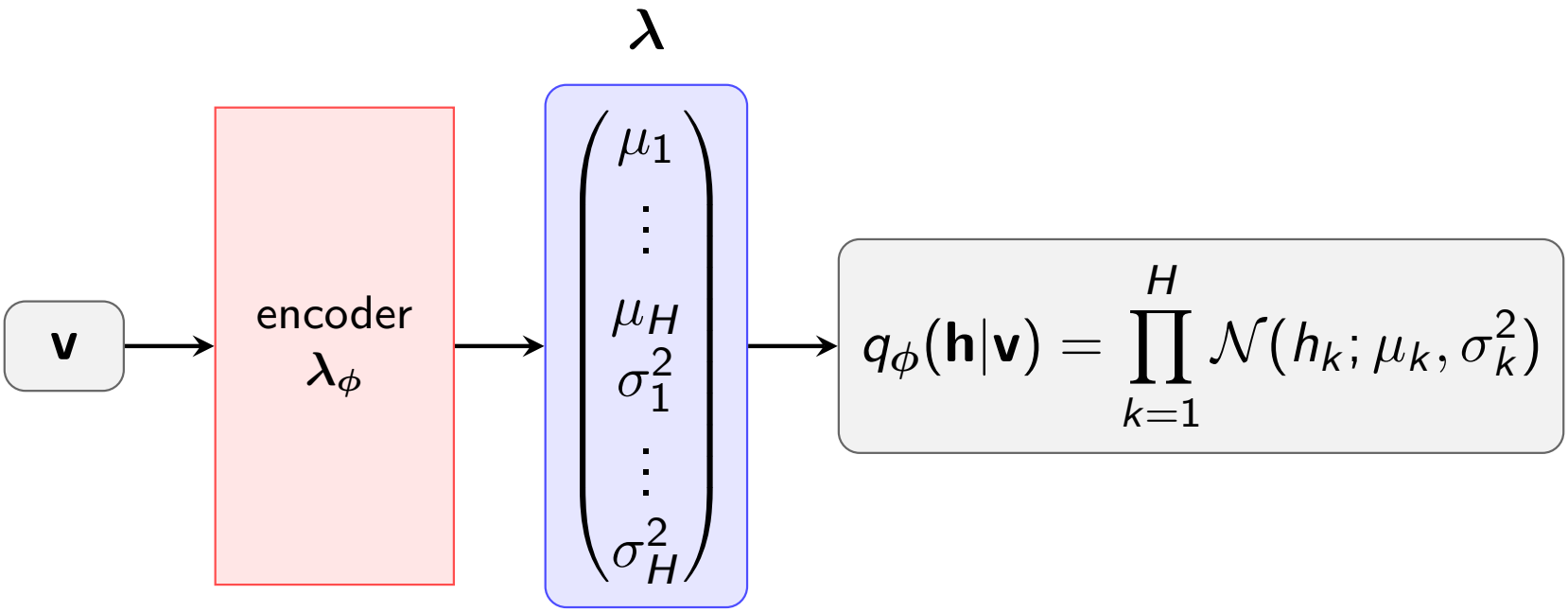
- n iterative optimisations
- parameters grow with n
- new $\mathbf{v}$: new optimisation



- one optimisation of $\sum_i \mathcal{L}_i(\boldsymbol{\theta}, \boldsymbol{\phi})$
- shared parameters $\boldsymbol{\phi}$, fixed size
- new $\mathbf{v}$: one forward pass $\boldsymbol{\lambda}_{\hat{\boldsymbol{\phi}}}(\mathbf{v})$

Amortisation (example)

- A popular choice, often used for variational autoencoders (see later): $\mathcal{Q}$ is the factorised Gaussian family (independence and parametric assumption).



- $\mu_k = \mu_k(\mathbf{v}; \phi)$ and $\sigma_k^2 = \sigma_k^2(\mathbf{v}; \phi)$ are nonlinear functions of $\mathbf{v}$ with parameters ϕ.

Amortisation gap

- After learning, we hope that $\boldsymbol{\lambda}_{\hat{\phi}}(\mathbf{v}_i) \approx \boldsymbol{\lambda}_i^*$ in (8).
- The ELBO lost by using the encoder is the amortisation gap

$$\mathcal{L}_i(\boldsymbol{\theta}, q_{\boldsymbol{\lambda}_i^*}) - \mathcal{L}_i(\boldsymbol{\theta}, q_{\hat{\phi}}) \geq 0 \tag{11}$$

- The gap is typically positive because
  - one shared ϕ has limited capacity
  - optimisation of ϕ is imperfect
- The gap is typically larger for test inputs $\mathbf{v}_{\text{test}}$ than for the training data $\mathbf{v}_1, \ldots, \mathbf{v}_n$.

Amortisation gap

- Example in two dimensions where $q(\mathbf{h}; \boldsymbol{\lambda})$ is Gaussian with parameters $\boldsymbol{\lambda} = (\mu_1, \mu_2)$.
- The contour plot shows $\mathcal{L}_i(\boldsymbol{\theta}, q_{\boldsymbol{\lambda}})$ as a function of $\boldsymbol{\lambda}$.
- The blue line shows the gradient ascent optimisation path when the ELBO is optimised without amortisation.
- The cyan diamond shows the amortised estimate $\boldsymbol{\lambda}_{\hat{\phi}}(\mathbf{v}_i)$.

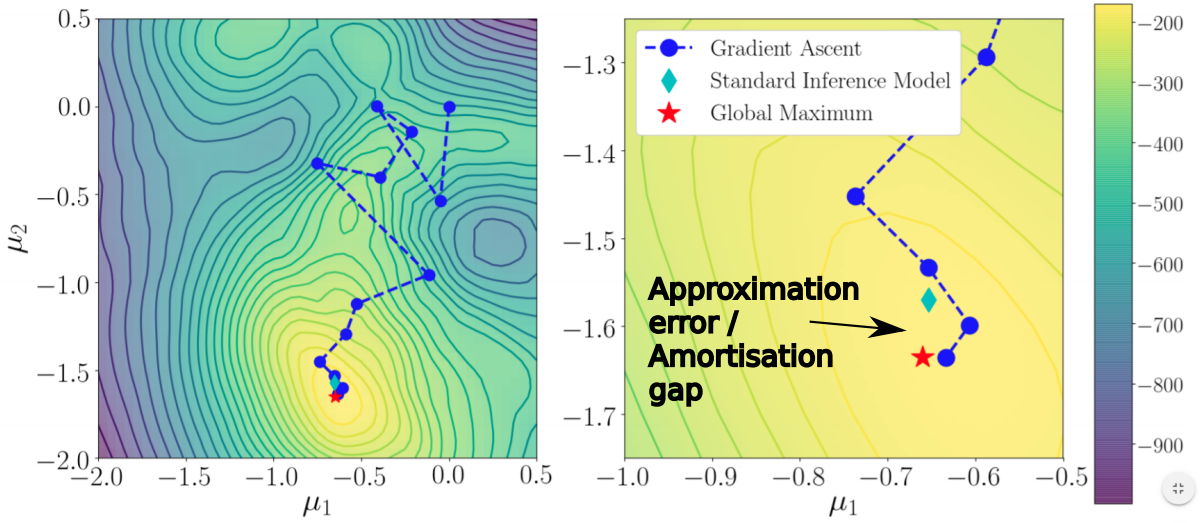


Figure 1 from Marino et al, *Iterative amortised inference*, ICML 2018, `https://arxiv.org/abs/1807.09356` (also proposes a method to reduce the gap)

Issue 2: Maximisation

- The optimisation problem is

$$\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\theta}, \boldsymbol{\phi}}{\operatorname{argmax}} \, \mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, \boldsymbol{\phi}) \tag{12}$$

  where

$$\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{i=1}^{n} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{h}_i|\mathbf{v}_i)} \left[\log \frac{p(\mathbf{v}_i, \mathbf{h}_i; \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\mathbf{h}_i|\mathbf{v}_i)} \right] \tag{13}$$

- We would like to solve it using gradient ascent.
- However:
  - We generally cannot compute the expectations in closed form.
  - The distribution $q_{\boldsymbol{\phi}}$ that defines the expectation depends on $\boldsymbol{\phi}$, so that we cannot pull $\nabla_{\boldsymbol{\phi}}$ inside the expectation (for $\nabla_{\boldsymbol{\theta}}$ we can).

Reparameterisation

- The gradient with respect to $\boldsymbol{\theta}$ is unproblematic: we can pull $\nabla_{\boldsymbol{\theta}}$ inside and approximate the expectation as a sample average with $\mathbf{h}_i \sim q_{\boldsymbol{\phi}}(\mathbf{h}_i|\mathbf{v}_i)$.
- For $\boldsymbol{\phi}$, this does not work: the samples themselves depend on $\boldsymbol{\phi}$, and we have to keep track of this dependency.
- For that, let us consider variational distributions $q_{\boldsymbol{\phi}}(\mathbf{h}|\mathbf{v})$ that can be obtained via a transformation of a random variable $\boldsymbol{\epsilon}$ whose distribution does not depend on $\boldsymbol{\phi}$.

$$\mathbf{h} \sim q_{\boldsymbol{\phi}}(\mathbf{h}|\mathbf{v}) \qquad \Longleftrightarrow \qquad \mathbf{h} = \mathbf{t}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}, \mathbf{v}), \quad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}) \tag{14}$$

- Examples:
  - $h \sim \mathcal{N}(h; \mu(\mathbf{v}), \sigma^2(\mathbf{v})) \Leftrightarrow h = \mu(\mathbf{v}) + \sigma(\mathbf{v})\epsilon$ with $\epsilon \sim \mathcal{N}(\epsilon; 0, 1)$.
  - Inverse transform sampling (see lecture on sampling)
  - Normalising flows (not covered in this course)
  - ...

Reparameterisation

- By the law of the unconscious statistician, we then obtain

$$\mathbb{E}_{q_\phi(\mathbf{h}_i|\mathbf{v}_i)}\left[\log \frac{p(\mathbf{v}_i, \mathbf{h}_i; \boldsymbol{\theta})}{q_\phi(\mathbf{h}_i|\mathbf{v}_i)}\right] = \mathbb{E}_{p(\boldsymbol{\epsilon}_i)}\left[\log \frac{p(\mathbf{v}_i, \mathbf{t}_\phi(\boldsymbol{\epsilon}_i, \mathbf{v}_i); \boldsymbol{\theta})}{q_\phi(\mathbf{t}_\phi(\boldsymbol{\epsilon}_i, \mathbf{v}_i)|\mathbf{v}_i)}\right] \quad (15)$$

- The distribution $p(\boldsymbol{\epsilon}_i)$ does not depend on ϕ; all ϕ-dependency is now inside the expectation. We can thus pull both $\nabla_{\boldsymbol{\theta}}$ and ∇_ϕ inside

$$\nabla_{\boldsymbol{\theta},\phi}\mathbb{E}_{q_\phi(\mathbf{h}_i|\mathbf{v}_i)}[\cdots] = \nabla_{\boldsymbol{\theta},\phi}\mathbb{E}_{p(\boldsymbol{\epsilon}_i)}[\cdots] = \mathbb{E}_{p(\boldsymbol{\epsilon}_i)}[\nabla_{\boldsymbol{\theta},\phi}\cdots]$$

- The gradient can then be computed via auto-differentiation.
- Requires that $\mathbf{t}_\phi(\boldsymbol{\epsilon}, \mathbf{v})$ is differentiable with respect to ϕ, which rules out e.g. discrete latents $\mathbf{h}$. An alternative for such cases is score function gradient estimation (not examinable).

Stochastic optimisation

- The gradient of $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, \boldsymbol{\phi})$ thus becomes

$$\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{i=1}^{n} \mathbb{E}_{p(\boldsymbol{\epsilon}_i)} \Bigg[\underbrace{\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \log \frac{p(\mathbf{v}_i, \mathbf{t}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}_i, \mathbf{v}_i); \boldsymbol{\theta})}{q_{\boldsymbol{\phi}}(\mathbf{t}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}_i, \mathbf{v}_i) | \mathbf{v}_i)}}_{\mathbf{g}_i(\boldsymbol{\epsilon}_i)} \Bigg] \quad (16)$$

- Evaluating it is expensive for large n:
  - the sum runs over all n data points,
  - each expectation $\mathbb{E}_{p(\boldsymbol{\epsilon}_i)}$ has to be approximated by a sample average (Monte Carlo integration) over S samples.
- This is not necessary. Stochastic optimisation only needs an unbiased estimate of the gradient: a random quantity that is cheap to compute and whose expectation equals (16).

We gloss over technical details here; for an introduction to stochastic optimisation, see *Introduction to Stochastic Search and Optimization* by James Spall.

Eq (16) can be manipulated to reduce the variance of the stochastic gradient, see Roeder et al, *Sticking the Landing: Simple, Lower-Variance Gradient Estimators for Variational Inference*, NeurIPS 2017.

Stochastic optimisation: minibatch gradient

- The gradient (16) is n times an average over the data of expectations over $\boldsymbol{\epsilon}_i$:

$$\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{i=1}^{n} \mathbb{E}_{p(\boldsymbol{\epsilon}_i)} \left[\mathbf{g}_i(\boldsymbol{\epsilon}_i) \right] = n \cdot \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{p(\boldsymbol{\epsilon}_i)} \left[\mathbf{g}_i(\boldsymbol{\epsilon}_i) \right]$$

- We can subsample the average with a random minibatch $\mathcal{B} \subset \{1, \ldots, n\}$, and approximate each expectation with S samples of $\boldsymbol{\epsilon}_i$:

$$\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, \boldsymbol{\phi}) \approx n \cdot \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \frac{1}{S} \sum_{s=1}^{S} \mathbf{g}_i(\boldsymbol{\epsilon}_i^{(s)}), \qquad \boldsymbol{\epsilon}_i^{(s)} \sim p(\boldsymbol{\epsilon}_i)$$

- Setting $S = 1$ (!) gives

$$\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, \boldsymbol{\phi}) \approx \frac{n}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \mathbf{g}_i(\boldsymbol{\epsilon}_i), \qquad \boldsymbol{\epsilon}_i \sim p(\boldsymbol{\epsilon}_i) \qquad (17)$$

- Noisy but unbiased: its expected value equals (16).

Program

1. Scalable generic variational learning of latent variable models
   - ELBO for iid data
   - Amortised variational inference
   - Reparameterisation and stochastic optimisation

2. Deep latent variable models and variational autoencoders

Program

1. Scalable generic variational learning of latent variable models

2. Deep latent variable models and variational autoencoders
   - Deep latent variable model
   - Variational autoencoder (VAE)
   - Gaussian and Bernoulli VAE
   - Learning the VAE

Deep directed graphical models

- Parametric directed graphical models are sets of pdfs/pmfs that factorise as

$$p(\mathbf{x};\boldsymbol{\theta}) = \prod_{k=1}^{d} p(x_k|\mathrm{pa}_k;\boldsymbol{\theta}) \tag{18}$$

where pa_k denotes the parents of x_k in a given directed acyclic graph (DAG).

- We say that the model is a deep directed graphical model if

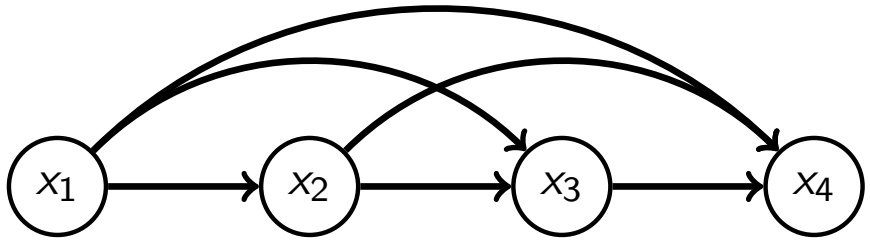
$$p(x_k|\mathrm{pa}_k;\boldsymbol{\theta}) = p(x_k;\boldsymbol{\eta}_k) \quad \text{with} \quad \boldsymbol{\eta}_k = \boldsymbol{\eta}_{\boldsymbol{\theta}}^k(\mathrm{pa}_k) \tag{19}$$

where $p(x_k;\boldsymbol{\eta})$ is a parametric model and $\boldsymbol{\eta}_{\boldsymbol{\theta}}^k(\mathrm{pa}_k)$ a parameterised nonlinear function (deep neural network) that maps the parents pa_k to the model-parameters $\boldsymbol{\eta}_k$.

Example

- Chain rule $p(\mathbf{x};\boldsymbol{\theta}) = \prod_{k=1}^{d} p(x_k|\mathrm{pre}_k;\boldsymbol{\theta})$ with

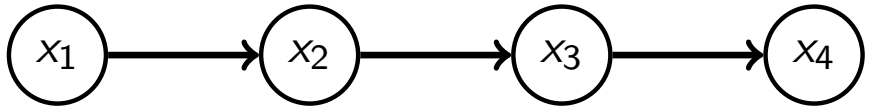
$$p(x_k|\mathrm{pre}_k;\boldsymbol{\theta}) = \mathcal{N}(x_k;\mu_k,\sigma_k^2), \qquad (\mu_k,\sigma_k^2) = \boldsymbol{\eta}_{\boldsymbol{\theta}}^k(\mathrm{pre}_k)$$



This is one of the autoregressive models from the slides *Basic Assumptions for Efficient Model Representation*.

- Markov chain $p(\mathbf{x};\boldsymbol{\theta}) = \prod_{k=1}^{d} p(x_k|x_{k-1};\boldsymbol{\theta})$ with

$$p(x_k|x_{k-1};\boldsymbol{\theta}) = \mathcal{N}(x_k;\mu_k,\sigma_k^2), \qquad (\mu_k,\sigma_k^2) = \boldsymbol{\eta}_{\boldsymbol{\theta}}^k(x_{k-1})$$



Deep latent variable model

- A deep (directed) latent variable model is a deep directed graphical model with latent variables.
- Often (but not always), they are models of the form

$$p(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) = p(\mathbf{v}|\mathbf{h}; \boldsymbol{\theta})p(\mathbf{h}) \tag{20}$$

where $p(\mathbf{h})$ does not depend on $\boldsymbol{\theta}$ and $p(\mathbf{v}|\mathbf{h}; \boldsymbol{\theta})$ is

$$p(\mathbf{v}|\mathbf{h}; \boldsymbol{\theta}) = \prod_{k=1}^{d} p(v_k|\check{\mathrm{pa}}_k, \mathbf{h}; \boldsymbol{\theta}) \tag{21}$$

with $\check{\mathrm{pa}}_k$ denoting the parents of v_k without $\mathbf{h}$.

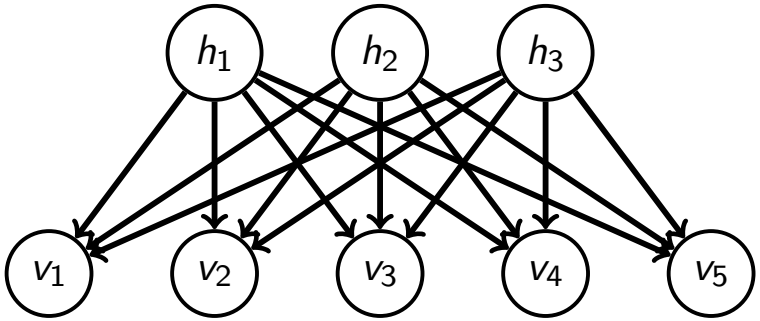
- The conditional is given by

$$p(v_k|\check{\mathrm{pa}}_k, \mathbf{h}; \boldsymbol{\theta}) = p(v_k; \boldsymbol{\eta}_k) \qquad \boldsymbol{\eta}_k = \boldsymbol{\eta}_{\boldsymbol{\theta}}^k(\check{\mathrm{pa}}_k, \mathbf{h}) \tag{22}$$

- Note: parameterised models $p(\mathbf{h}; \boldsymbol{\theta})$ may also be used.

Variational autoencoder (VAE): model

Reconsider the directed acyclic graph for FA and ICA:



- The visibles $\mathbf{v} = (v_1, \ldots, v_d)$ are independent given the latents $\mathbf{h} = (h_1, \ldots, h_H)$. Different assumptions on $p(v_k|\mathbf{h})$ and $p(\mathbf{h})$ give different methods, e.g. FA and ICA.
- The VAE uses a deep latent variable model with

$$p(\mathbf{h}) = \mathcal{N}(\mathbf{h}; \mathbf{0}, \mathbf{I}), \quad p(\mathbf{v}|\mathbf{h}; \boldsymbol{\theta}) = \prod_{k=1}^{d} p(v_k; \boldsymbol{\eta}_k), \quad \boldsymbol{\eta}_k = \boldsymbol{\eta}_{\boldsymbol{\theta}}^k(\mathbf{h})$$

- $\boldsymbol{\eta}_{\boldsymbol{\theta}}^k(\mathbf{h})$ is a nonlinear function (neural network) with parameters $\boldsymbol{\theta}$. Typically $H < d$.

Variational autoencoder (VAE): encoder and decoder

- The variational distribution is the amortised factorised Gaussian from before:

$$q_\phi(\mathbf{h}|\mathbf{v}) = \prod_{k=1}^{H} \mathcal{N}(h_k; \mu_k(\mathbf{v}), \sigma_k^2(\mathbf{v}))$$

  with $(\mu_1, \ldots, \mu_H, \sigma_1^2, \ldots, \sigma_H^2) = \boldsymbol{\lambda}_\phi(\mathbf{v})$. ($\phi$-dependency of $\mu_k(\mathbf{v}), \sigma_k^2(\mathbf{v})$ is suppressed.)
- VAE: the model $p(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta})$ and the variational distribution $q_\phi(\mathbf{h}|\mathbf{v})$, with $\boldsymbol{\theta}$ and ϕ learned jointly by maximising $\mathcal{L}_\mathcal{D}(\boldsymbol{\theta}, \phi)$.
- Names:
  - $q_\phi(\mathbf{h}|\mathbf{v})$: encoder, inference model, or recognition model
  - $p(\mathbf{v}|\mathbf{h}; \boldsymbol{\theta})$: decoder or generative model

  The names may refer to the distribution or to the network that maps to its parameters.

Gaussian VAE

- The family for $p(v_k; \boldsymbol{\eta}_k)$ is chosen depending on the data. It may be simple, but $\boldsymbol{\eta}_k = \boldsymbol{\eta}_{\boldsymbol{\theta}}^k(\mathbf{h})$ is a nonlinear function of $\mathbf{h}$, which gives a flexible class of models.
- The Gaussian VAE (for $v_k \in \mathbb{R}$) is obtained for

$$p(v_k|\mathbf{h}; \boldsymbol{\theta}) = \mathcal{N}(v_k; m_k, s_k^2) \qquad (m_k, s_k^2) = \boldsymbol{\eta}_{\boldsymbol{\theta}}^k(\mathbf{h}) \qquad (23)$$

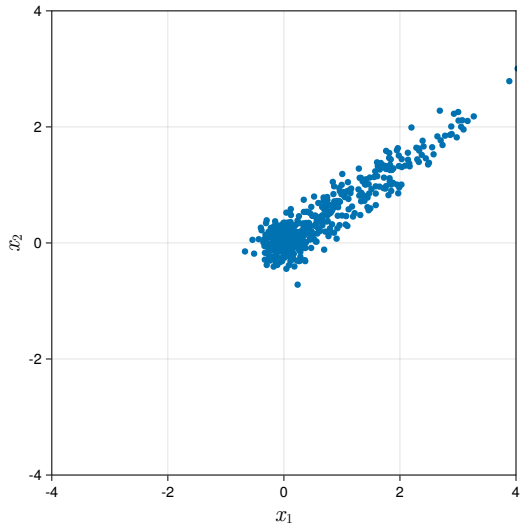
- Generative model $p(\mathbf{v}|\mathbf{h}; \boldsymbol{\theta})$ equivalent to

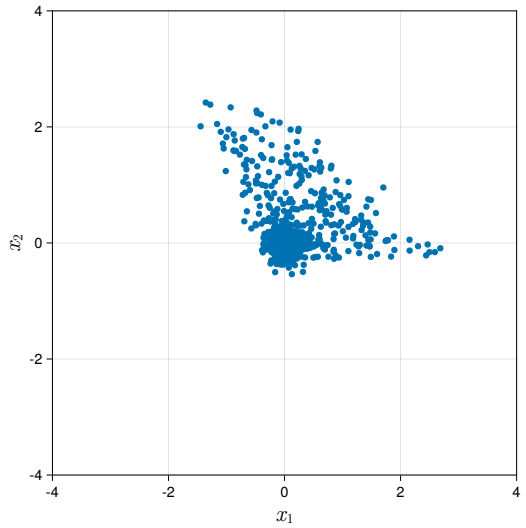
$$\mathbf{v} = \begin{pmatrix} m_1(\mathbf{h}) \\ \vdots \\ m_d(\mathbf{h}) \end{pmatrix} + \begin{pmatrix} s_1(\mathbf{h}) & & \\ & \ddots & \\ & & s_d(\mathbf{h}) \end{pmatrix} \mathbf{n}, \quad \mathbf{n} \sim \mathcal{N}(\mathbf{n}; \mathbf{0}, \mathbf{I})$$

- FA obtained for $\mathbf{m} = (m_1, \ldots, m_d)^\top = \mathbf{F}\mathbf{h} + \mathbf{c}$ and $s_k^2 = \Psi_k$: the Gaussian VAE is a nonlinear generalisation of FA.

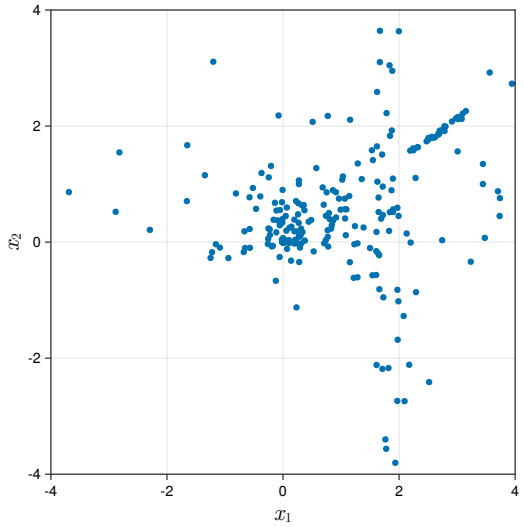
Example: Gaussian VAE

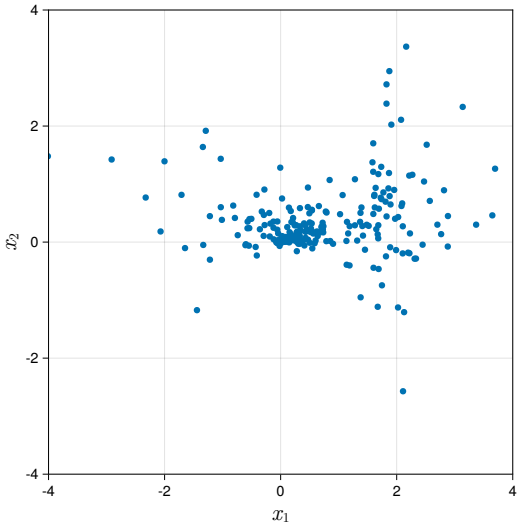
Nonlinear mean function (NN with random weights and ReLu), constant variance:





Nonlinear mean and variance functions:





Bernoulli VAE

- The Bernoulli VAE with $v_k \in \{0, 1\}$ is obtained for

$$p(v_k|\mathbf{h}; \boldsymbol{\theta}) = p_k^{v_k}(1 - p_k)^{(1-v_k)} \qquad p_k = \eta_{\boldsymbol{\theta}}^k(\mathbf{h}) \tag{24}$$

- This is often also used for $v_k \in [0, 1]$. While the ELBO can be evaluated, it is formally wrong since v_k is not binary.
- For $v_k \in [0, 1]$, use the so-called continuous Bernoulli distribution or the beta distribution instead.

  (see Loaiza-Ganem and Cunningham, *The continuous Bernoulli: fixing a pervasive error in variational autoencoders*, NeurIPS 2019)

VAE: objective

- We use the following form of the ELBO

$$\mathcal{L}_i = \mathbb{E}_{q_\phi(\mathbf{h}_i|\mathbf{v}_i)}\left[\log p(\mathbf{v}_i|\mathbf{h}_i;\boldsymbol{\theta})\right] - \mathrm{KL}(q_\phi(\mathbf{h}_i|\mathbf{v}_i)||\mathcal{N}(\mathbf{h}_i;\mathbf{0},\mathbf{I}))$$

- First term: reconstruction. For a Gaussian VAE with fixed variances $s_k^2 = s^2$ and parameterised mean function $\mathbf{m}_{\boldsymbol{\theta}}(\mathbf{h})$:

$$\mathbb{E}_{q_\phi(\mathbf{h}_i|\mathbf{v}_i)}\left[\log p(\mathbf{v}_i|\mathbf{h}_i;\boldsymbol{\theta})\right] = -\frac{1}{2s^2}\mathbb{E}_{q_\phi(\mathbf{h}_i|\mathbf{v}_i)}\left[\|\mathbf{v}_i - \mathbf{m}_{\boldsymbol{\theta}}(\mathbf{h}_i)\|^2\right] + \mathrm{c}$$

  This is the expected error when $\mathbf{v}_i$ is encoded into $\mathbf{h}_i \sim q_\phi(\mathbf{h}_i|\mathbf{v}_i)$ and decoded into $\mathbf{m}_{\boldsymbol{\theta}}(\mathbf{h}_i)$: hence "autoencoder"; "variational" since the encoder is the variational distribution.

- Second term: regularisation. It keeps $q_\phi(\mathbf{h}_i|\mathbf{v}_i)$ close to the prior $p(\mathbf{h}) = \mathcal{N}(\mathbf{h};\mathbf{0},\mathbf{I})$.

VAE: objective

- Closed-form KL for the factorised Gaussian:

$$\mathrm{KL}(q_\phi(\mathbf{h}_i|\mathbf{v}_i)||\mathcal{N}(\mathbf{h}_i;\mathbf{0},\mathbf{I})) = \frac{1}{2}\sum_{k=1}^{H}\left(\sigma_k^2(\mathbf{v}_i) + \mu_k^2(\mathbf{v}_i) - 1 - \log\sigma_k^2(\mathbf{v}_i)\right)$$

- Conditional independence in $p(\mathbf{v}|\mathbf{h};\boldsymbol{\theta})$, with v_{ik} the k-th element of $\mathbf{v}_i$:

$$\mathbb{E}_{q_\phi(\mathbf{h}_i|\mathbf{v}_i)}\left[\log p(\mathbf{v}_i|\mathbf{h}_i;\boldsymbol{\theta})\right] = \sum_{k=1}^{d}\mathbb{E}_{q_\phi(\mathbf{h}_i|\mathbf{v}_i)}\left[\log p(v_{ik};\boldsymbol{\eta}_\theta^k(\mathbf{h}_i))\right]$$

- We thus have for the VAE:

$$\mathcal{L}_i(\boldsymbol{\theta},\boldsymbol{\phi}) = \sum_{k=1}^{d}\mathbb{E}_{q_\phi(\mathbf{h}_i|\mathbf{v}_i)}\left[\log p(v_{ik};\boldsymbol{\eta}_\theta^k(\mathbf{h}_i))\right] + \frac{1}{2}\sum_{k=1}^{H}\left(1 + \log\sigma_k^2(\mathbf{v}_i) - \sigma_k^2(\mathbf{v}_i) - \mu_k^2(\mathbf{v}_i)\right) \quad (25)$$

VAE: reparameterisation and stochastic optimisation

- We maximise $\sum_{i=1}^{n} \mathcal{L}_i(\boldsymbol{\theta}, \boldsymbol{\phi})$ with respect to $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.
- Reparameterisation of the factorised Gaussian: sampling $\mathbf{h}_i \sim q_{\boldsymbol{\phi}}(\mathbf{h}_i|\mathbf{v}_i)$ can be done via

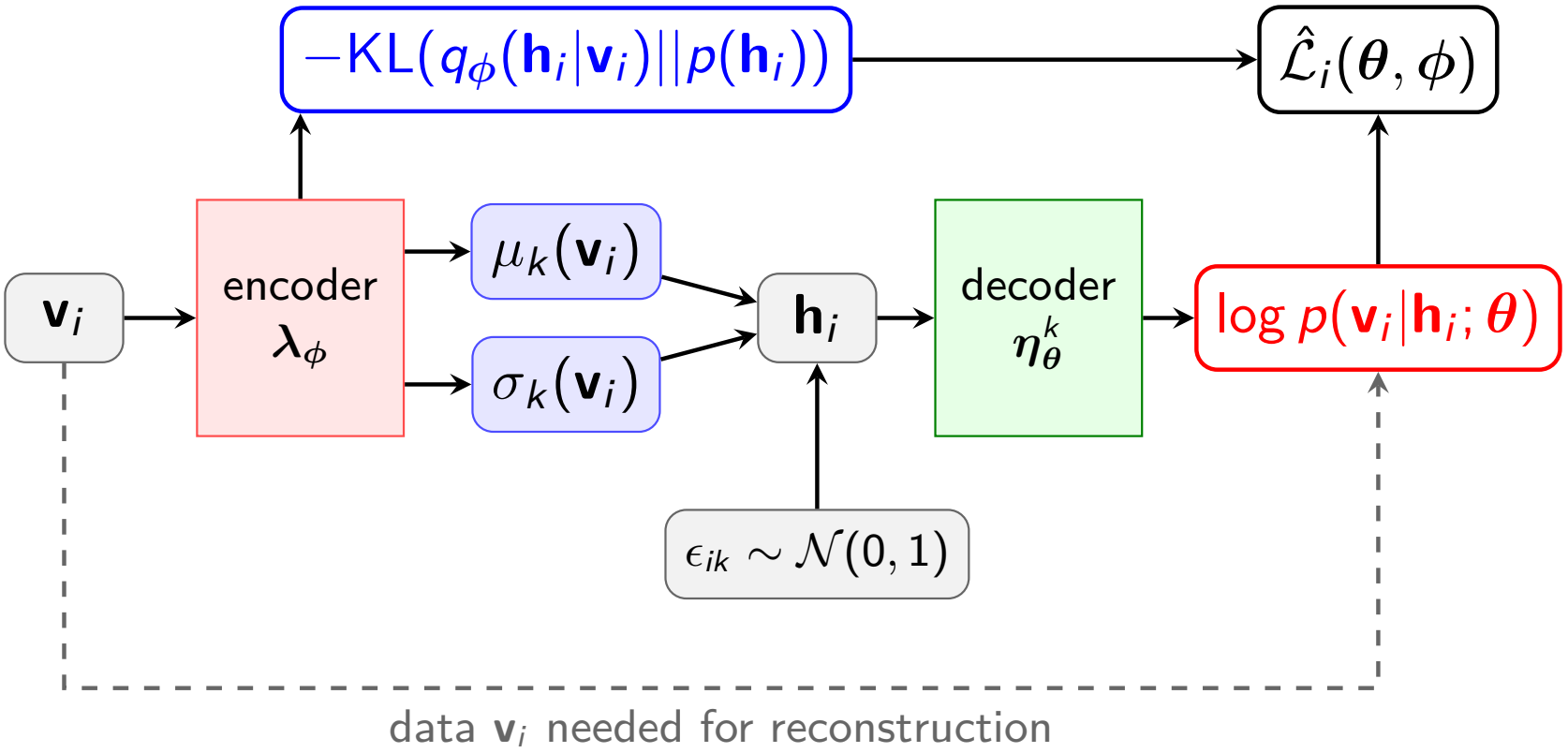
$$h_{ik} = \mu_k(\mathbf{v}_i) + \sigma_k(\mathbf{v}_i)\epsilon_{ik}, \quad \epsilon_{ik} \sim \mathcal{N}(\epsilon_{ik}; 0, 1), \quad k = 1, \ldots, H$$

- One-sample estimate of $\mathcal{L}_i$, with $\mathbf{h}_i = (h_{i1}, \ldots, h_{iH})$ as above:

$$\hat{\mathcal{L}}_i(\boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{k=1}^{d} \log p(v_{ik}; \eta_{\boldsymbol{\theta}}^k(\mathbf{h}_i)) + \frac{1}{2} \sum_{k=1}^{H} \left(1 + \log \sigma_k^2(\mathbf{v}_i) - \sigma_k^2(\mathbf{v}_i) - \mu_k^2(\mathbf{v}_i)\right)$$

This is a noisy but unbiased estimate of $\mathcal{L}_i(\boldsymbol{\theta}, \boldsymbol{\phi})$.

VAE: reparameterisation and stochastic optimisation



- Reparameterisation trick: $h_{ik} = \mu_k(\mathbf{v}_i) + \sigma_k(\mathbf{v}_i)\epsilon_{ik}$. The randomness is in $\boldsymbol{\epsilon}_i$ only, so $\hat{\mathcal{L}}_i$ is a differentiable function of $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.
- $\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}}\hat{\mathcal{L}}_i$ by backpropagation through this graph.

VAE: training algorithm

Stochastic gradient ascent, repeated until convergence:

1. Sample a minibatch $\mathcal{B} \subset \{1, \ldots, n\}$.
2. For each $i \in \mathcal{B}$:
   - encode: $(\mu_1(\mathbf{v}_i), \ldots, \sigma_H^2(\mathbf{v}_i)) = \boldsymbol{\lambda}_\phi(\mathbf{v}_i)$
   - sample: $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\boldsymbol{\epsilon}_i; \mathbf{0}, \mathbf{I})$ and set $h_{ik} = \mu_k(\mathbf{v}_i) + \sigma_k(\mathbf{v}_i)\epsilon_{ik}$ (reparameterisation trick)
   - decode: $\boldsymbol{\eta}_{\boldsymbol{\theta}}^k(\mathbf{h}_i)$ for $k = 1, \ldots, d$, and evaluate $\hat{\mathcal{L}}_i(\boldsymbol{\theta}, \boldsymbol{\phi})$
3. Compute the minibatch gradient by auto-differentiation:

$$\hat{\mathbf{g}} = \frac{n}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_{\boldsymbol{\theta}, \boldsymbol{\phi}} \hat{\mathcal{L}}_i(\boldsymbol{\theta}, \boldsymbol{\phi})$$

   It is a noisy but unbiased estimate of $\nabla_{\boldsymbol{\theta}, \boldsymbol{\phi}} \mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta}, \boldsymbol{\phi})$ (see *Stochastic optimisation*).
4. Update $(\boldsymbol{\theta}, \boldsymbol{\phi}) \leftarrow (\boldsymbol{\theta}, \boldsymbol{\phi}) + \alpha\, \hat{\mathbf{g}}$ with step size α.

Program recap

1. Scalable generic variational learning of latent variable models
   - ELBO for iid data
   - Amortised variational inference
   - Reparameterisation and stochastic optimisation

2. Deep latent variable models and variational autoencoders
   - Deep latent variable model
   - Variational autoencoder (VAE)
   - Gaussian and Bernoulli VAE
   - Learning the VAE